Optimizing the Multilayer Feed-Forward Artificial Neural Networks Architecture and Training Parameters using Genetic Algorithm

Osman Ahmed Abdalla
University of Tabuk
Faculty of Computers and Information Technology
Tabuk, KSA

Abdelrahman Osman Elfaki
University of Tabuk
Faculty of Computers and Information Technology
Tabuk, KSA

Yahya Mohammed AlMurtadha
University of Tabuk
Faculty of Computers and Information Technology
Tabuk, KSA

ABSTRACT
Determination of optimum feed forward artificial neural network (ANN) design and training parameters is an extremely important mission. It is a challenging and daunting task to find an ANN design, which is effective and accurate. This paper presents a new methodology for the optimization of ANN parameters as it introduces a process of training ANN which is effective and less human-dependent. The derived ANN achieves satisfactory performance and solves the time-consuming task of training process. A Genetic Algorithm (GA) has been used to optimize training algorithms, network architecture (i.e. number of hidden layer and neurons per layer), activation functions, initial weight, learning rate, momentum rate, and number of iterations. The preliminary result of the proposed approach has indicated that the new methodology can optimize designing and training parameters precisely, and resulting in ANN where satisfactory performance.

General Terms
Neural network, Feed forward networks, genetic algorithm, neural networks parameters.

Keywords
Optimizing neural networks, neural networks and genetic algorithm

1. INTRODUCTION
Determination of artificial neural network (ANN) parameters for Design and training process for optimization ability is an extremely important task. This requires ANN design that is more accurate and less human-dependent, performs satisfactorily, and able to solve the time-consuming task of training process. This paper looks into determining ANN parameters and improving the efficiency of network performance by adopting genetic algorithm (GA).

Generally, the successful application of ANN for the purpose of prediction and modeling in science and engineering domains is tremendously affected by these consequent factors: network architecture (i.e. number of hidden layers and number of neurons in each hidden layer), training algorithms (i.e. gradient descent, quasi-Newton, conjugate gradient, Levenberg-Marquardt, resilient back propagation algorithm, etc), activation functions (i.e. Log-sigmoid, Softmax, Linear, etc), learning rate, momentum rate, and number of iterations.

ANN [1,2, 3], in particular, is utilized for modeling and prediction purpose due to non-linearity, time variability, and difficulty in inferring input-output mapping, whereas GA is a search algorithm for optimization, based on genetic and evolution principles [4,5,6]. Current literatures on ANN show that the determination of optimal design and training parameters is the major obstacles for their daily usage, accuracy, and effectiveness.

The number of hidden layers and number of neurons in each layer have significant impact in network training performance, training time and its generalization abilities. Furthermore, the use of huge number of neurons in hidden layer may over fit the data, which may cause the loss of generalization capability of network. Besides, small number of neurons in hidden layer may under fit the data; subsequently the network may not be able to learn.

The main objective of this work is to design ANN with satisfactory performance and less human-dependent, and able to solve the time-consuming task of training process.

2. ARTIFICIAL NEURAL NETWORK
The ANN has been used to emulate the human decision and prediction ability. In recent years, ANN technique has been applied successfully in many fields, such as automotive [7], banking [8], electronics [9], financial [10], industrial [11], telecommunications [12], oil and gas [13], and robotics [14].

ANN consists of several layers, which are input, hidden, and output layers. Feed-forward is the most popular neural network type while back propagation algorithm is widely used as training algorithm.

ANN modeling has three commonly used stage operators, which are training, validation, and testing network. Training stage involves adjustment to the connection (weight) that exists between neurons. To achieve this, a number of data set (usually huge) are used. Validation is usually used to tune parameters, i.e., architecture, for example to choose the number of hidden layers. Eventually testing stage is used to ensure the well network generalization.

.

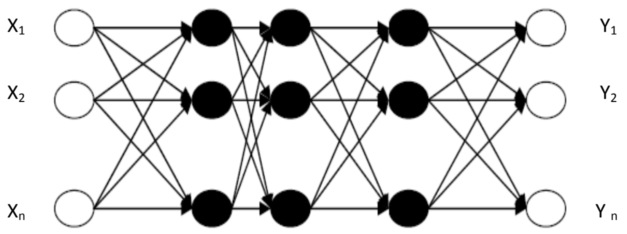


Fig. 1: Structure of neural network of n inputs, n hidden, and n outputs layers

3. GENETIC ALGORITHM

GA is a combinatorial optimization technique and general purpose optimization method based on Darwin Theory of Evolution that searches for an optimal/near value of a complex objective function by simulation of the natural evolutionary process. GA has been successfully used in a wide variety of problem domains (Goldberg, 1989).

In brief, GA consists of three basic operators: selection, crossover, and mutation. The algorithm starts with a set of solutions to the problem being examined; the solution set (represented by chromosomes in GA) is called the population.

Crossover operation is used to obtain a new solution by combining different chromosomes to generate new better offspring; as well as new solution by altering existing member of the population, this operation is called mutation. These operations can be represented as shown in Figure 2 and 3.

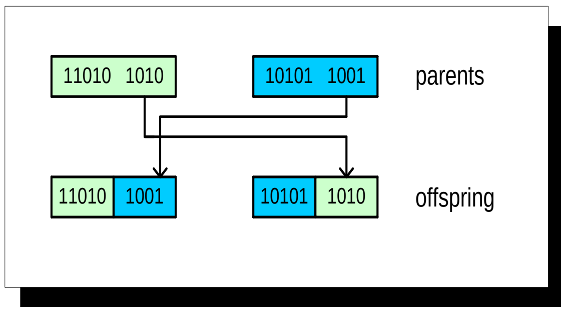


Fig2: Crossover operation

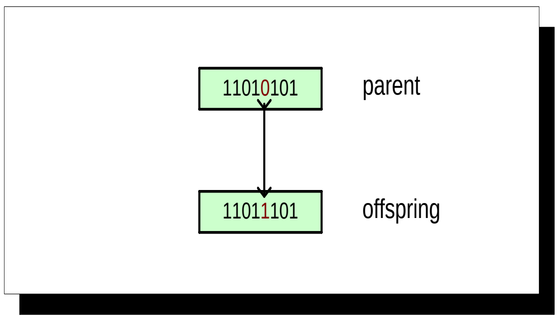


Fig 3: Mutation operation

4. RELATED WORK

In recent years, many works in science and engineering domains have been successfully applied, but there is still no general method to design ANN. In this paper, we will mainly concern with the optimization of ANN designing parameters in order to get fully-automated ANN design through GA, numerical, and statistical methods. In this aspect there are several works that have been done; which can be roughly summarized as follows:

Genetic algorithm has been adopted in the neural network to avoid inadequate current trial-and-error approach and to determine a set of initial process parameters for injection moulding [15]. To map the complex and nonlinear relationships among the parameters involved in the initial process parameter of injection moulding, neural network (NN) was developed and implemented and genetic algorithm (GA) was applied to search and identify a set of optimal/near optimal process parameters for injection moulding. The developed NN-GA approach consists of two parts: an NN prediction and a GA part. First, an initial population is randomly generated, which contains a number of sets of initial process parameters. Then the strings stored in it are individually fed into a NN-based prediction unit for the quality prediction of moulded parts. The developed hybrid NN-GA system has significantly reduced the time required to generate initial process parameters for injection moulding in comparison to the mould flow simulation for injection moulding, where it takes less than 2 minutes including the time for user input to obtain a set of initial process parameters corresponding to an input problem.

Another application of genetic algorithm is to search for optimal hidden layer architectures, connectivity, and training parameters for ANN for predicting community acquired pneumonia among patients with respiratory complaints. Feed-forward ANN that uses back propagation algorithm with 35 nodes in the input layer, one node in the output layer, and between 0 and 15 nodes in each of 0, 1, or 2 hidden layers, is determined by the developed genetic algorithm. Neural network structure and training parameters are represented by haploid chromosomes consisting of ‘‘genes’’ of binary numbers. Each chromosome has five genes. The first two genes are 4-bit binary numbers, representing the number of nodes in the first and second hidden layers of the network, each of which could range from 0 to 15 nodes. The third and fourth genes are 2-bit binary numbers, representing the learning rate and momentum with which the network has been trained, which each could assume discrete values of 0.01, 0.05, 0.1, or 0.5. The fifth gene is a 1-bit binary number, representing whether implicit within-layer connectivity using the competition algorithm [16].

A feed-forward neural network model for fault detection NN of a deep-trough hydroponic system and a predictive modeling NN system of a similar hydroponic system has also been developed. In those two models, a genetic algorithm (GA) is used to encode NN topologies, activation function and training function in the field of biological engineering, and more specifically in modern hydroponic plant production systems [17]. A back propagation for the training of the NN is used as minimization algorithm. The NN model is used to detect and diagnose possible faults in the operation of the hydroponic system by measuring conditions of the system in real time. The second NN model is used to predict one-step-ahead values of the pH, and the electrical conductivity of a modern hydroponic system for plant production. The GA is used to determine the minimization algorithm used by the back propagation training algorithm, network architecture, types of activation functions of the hidden nodes and of the output nodes. Both models gave satisfactory performance, and the GA system has successfully replaced the problematic trial-and-error method, previously used in this task.

5. METHODOLOGY

This paper addresses the ANN designing problems that occurred when using a cumbersome trial-and-error procedure by adopting a methodology based on GA method. This involves optimally designing the ANN parameters which include, network architecture, activation function, training algorithm, learning rate, momentum rate and number of epoch as depicted in Figure 4.

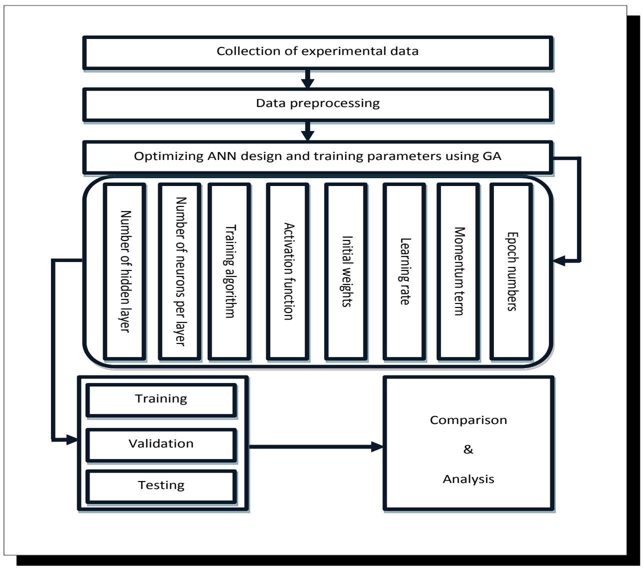


Fig 4: Methodology steps

5.1 Data Collection

Two different dataset were collected and used to obtain ANN-GA approach. One from PETRONAS Penapisan (Melaka) Sdn Bhd from Jan. to Feb. 2007. The second dataset is XOR standard problem dataset.

A 500-data set from PETRONAS Penapisan (Melaka) Sdn Bhd, and 4-dataset of XOR problem have been used to develop the new proposed model.

5.2 Pretreatment and analysis of the dataset

All data were crosschecked visually and statistically to ensure accuracy and validity of the input data. New simple Matlab function was used to remove non-number values and to apply a cut-off-error percentage method; this method has been used to remove invalid observations and to avoid the outlier error [19, 20].

Furthermore, in order to achieve best performance of ANN model, the experimental dataset were further randomly partitioned into three different sets: training, validation, and testing using the most popular ratio 3-1-1.

5.3 Preparations and scaling of the dataset

The training, validation and testing dataset were scaled to the range of (0–1) using the modified MATLAB functions ‘premnmx’ and ‘tramnmx’. The following equation was used for the purpose:

$$x_{ni} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \qquad (1)$$

Where x_i is the real-world input value, x_{ni} is the scaled input value of the real-world input value x_i and x_{min} and x_{max} are the corresponding minimum and maximum values of the unscaled dataset.

The network predicted values, which were in the range of (0–1), were transformed to real-world values using the modified MATLAB function ‘postmnmx’. The equation below was used for the purpose:

$$x_i = x_{ni}(x_{max} - x_{min}) + x_{min} \qquad (2)$$

5.4 ANN development model

Three feed-forward neural network models in the fields of petroleum and energy have been developed. The first NN model is used to predict iso-pentane (iC5) and normal pentane (nC5) of debutanizer CRU, the input layer contains three input neurons and represents the input variables, which are temperature, reflux flow, and flow rate. The output layer contains two neurons (iC5 and nC5). Second NN model is used to predict the output of XOR problem, this model has two input neurons and one output neurons.

5.5 ANN-GA combination

The combination of ANN and GA technique is a powerful method for modeling and optimization purposes. In this paper, the GA is applied to obtain the optimal ANN design and training parameters.

5.5.1 GA encoding

In this work, NN design and training parameters are represented by weak specification encoding scheme, which is the way of encoding the several possible chromosomes of the NN into specific genotypes. A chromosome, in our case, consists of the, training algorithm, NN architecture, and its activation functions in the hidden and output nodes, the learning rate, momentum rate, and iteration number. A genotype is a sequence of bits (0 or 1) with a specific constant length. Each genotype corresponds to a unique chromosome.

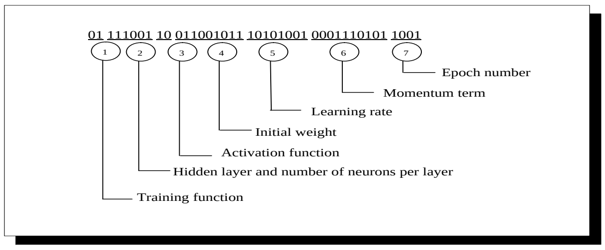


Fig 5: GA encoding to represent ANN parameters

5.5.2 Representation of the minimization algorithm

The back propagation [22] has been very successful in solving several problems as training algorithm for the purpose of adjustment of the connection weights of the networks throughout the training process. In general, there are four different minimization algorithms considered by the proposed GA encoding: quasi-Newton, steepest descent, conjugate gradient, and Levenberg–Marquardt algorithms. There is no general method concerning to which algorithm is appropriate for which application. Based on our knowledge, most of the previous works used trial-and-error approach and user experience to select the best training algorithm. These four algorithms can be represented as shown in Table1.

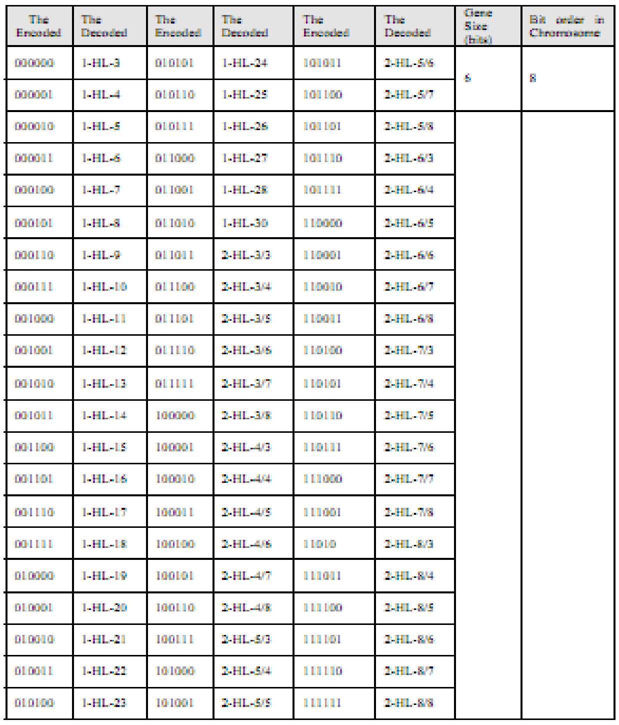
Table 1: Binary encoding of training algorithm of the multilayer feed-forward ANN

The Encoded	The Decoded	Gene Size (bits)	Bit order in Chromosome
00	Steepest Descent	2	1-2
01	Quasi-Newton		
10	Levenberg–Marquardt		
11	Conjugate Gradient		

5.5.3 Representation of the network architecture

The 64 potential network architectures of one-hidden-layer (1-HL) and two-hidden-layer (2-HL) networks were represented by the next six binary entries of the string. Table 2.

Table 2: Binary encoding of ANN architecture



The Encoded	The Decoded	The Encoded	The Decoded	The Encoded	The Decoded	Gene Size (bits)	Bit order in Chromosome
000000	1-HL-3	010101	1-HL-24	101011	2-HL-5/6	6	8
000001	1-HL-4	010110	1-HL-25	101100	2-HL-5/7		
000010	1-HL-5	010111	1-HL-26	101101	2-HL-5/8		
000011	1-HL-6	011000	1-HL-27	101110	2-HL-6/3		
000100	1-HL-7	011001	1-HL-28	101111	2-HL-6/4		
000101	1-HL-8	011010	1-HL-30	110000	2-HL-6/5		
000110	1-HL-9	011011	2-HL-3/3	110001	2-HL-6/6		
000111	1-HL-10	011100	2-HL-3/4	110010	2-HL-6/7		
001000	1-HL-11	011101	2-HL-3/5	110011	2-HL-6/8		
001001	1-HL-12	011110	2-HL-3/6	110100	2-HL-7/3		
001010	1-HL-13	011111	2-HL-3/7	110101	2-HL-7/4		
001011	1-HL-14	100000	2-HL-3/8	110110	2-HL-7/5		
001100	1-HL-15	100001	2-HL-4/3	110111	2-HL-7/6		
001101	1-HL-16	100010	2-HL-4/4	111000	2-HL-7/7		
001110	1-HL-17	100011	2-HL-4/5	111001	2-HL-7/8		
001111	1-HL-18	100100	2-HL-4/6	11010	2-HL-8/3		
010000	1-HL-19	100101	2-HL-4/7	111011	2-HL-8/4		
010001	1-HL-20	100110	2-HL-4/8	111100	2-HL-8/5		
010010	1-HL-21	100111	2-HL-5/3	111101	2-HL-8/6		
010011	1-HL-22	101000	2-HL-5/4	111110	2-HL-8/7		
010100	1-HL-23	101001	2-HL-5/5	111111	2-HL-8/8		

1-HL-x: one-hidden-layer with x hidden neurons; 2-HL-x/y: two-hidden-layer with x hidden neurons in the first hidden layer and y hidden neurons in second hidden layer.

5.5.4 Representation of the activation functions

In this work three activation functions were considered for the hidden nodes of the NN:

1. The logistic function
2. The tangent sigmoid function.
3. The linear function

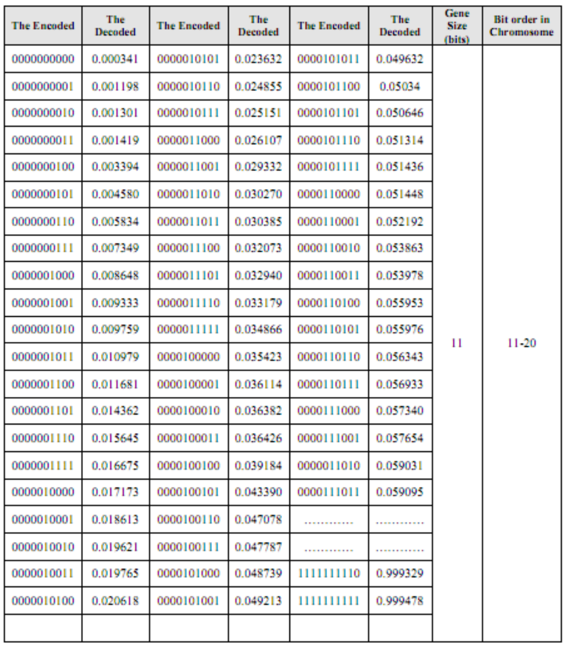
Table 3: Binary encoding of activation functions of hidden and output nodes of the ANN

The Encoded	The Decoded		Gene Size (bits)	Bit order in Chromosome
	Hidden neuron's Activation function	Output neuron's Activation function		
00	Logistic	Logistic	2	9-10
01	Logistic	Pure Linear		
10	Tangent sigmoid	Pure Linear		
11	Tangent sigmoid	Logistic		

5.5.5 Representation of initial weight

The 11th and 20th genes represent the discrete value discrete value of initial weight with which the network was trained. Table 4.

Table 4: Binary encoding of learning rate value of the ANN

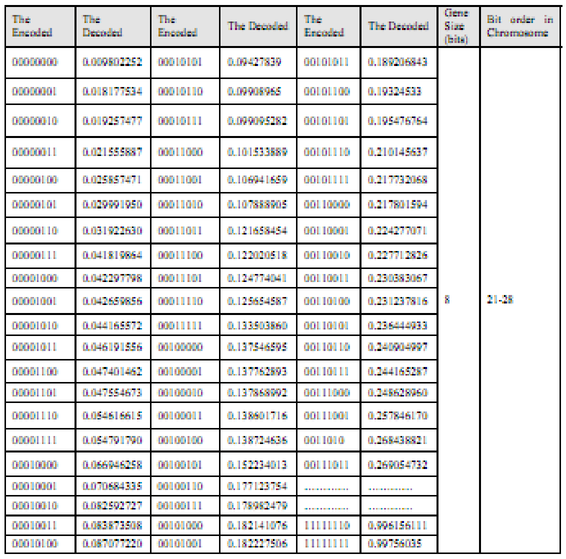


The Encoded	The Decoded	The Encoded	The Decoded	The Encoded	The Decoded	Gene Size (bits)	Bit order in Chromosome
0000000000	0.000341	0000010101	0.023632	0000101011	0.049632	11	11-20
0000000001	0.001198	0000010110	0.024855	0000101100	0.05034		
0000000010	0.001301	0000010111	0.025151	0000101101	0.050646		
0000000011	0.001419	0000011000	0.026107	0000101110	0.051314		
0000000100	0.003394	0000011001	0.029332	0000101111	0.051436		
0000000101	0.004580	0000011010	0.030270	0000110000	0.051448		
0000000110	0.005834	0000011011	0.030385	0000110001	0.052192		
0000000111	0.007349	0000011100	0.032073	0000110010	0.053863		
0000001000	0.008648	0000011101	0.032940	0000110011	0.053978		
0000001001	0.009333	0000011110	0.033179	0000110100	0.055953		
0000001010	0.009759	0000011111	0.034866	0000110101	0.055976		
0000001011	0.010979	0000100000	0.035423	0000110110	0.056343		
0000001100	0.011681	0000100001	0.036114	0000110111	0.056933		
0000001101	0.014362	0000100010	0.036382	0000111000	0.057340		
0000001110	0.015645	0000100011	0.036426	0000111001	0.057654		
0000001111	0.016675	0000100100	0.039184	0000011010	0.059031		
0000010000	0.017173	0000100101	0.043390	0000111011	0.059095		
0000010001	0.018613	0000100110	0.047078	…………	…………		
0000010010	0.019621	0000100111	0.047787	…………	…………		
0000010011	0.019765	0000101000	0.048739	1111111110	0.999329		
0000010100	0.020618	0000101001	0.049213	1111111111	0.999478		

5.5.6 Representation of the learning rate

The 21th and 28th genes represent the discrete value of learning rate with which the network was trained. Table 5.

Table 5: Binary encoding of learning rate value of the ANN

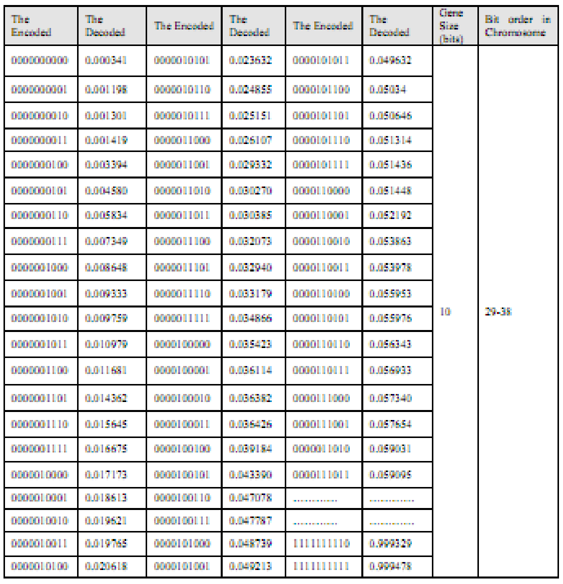


The Encoded	The Decoded	The Encoded	The Decoded	The Encoded	The Decoded	Gene Size (bits)	Bit order in Chromosome
00000000	0.009802252	00010101	0.09427839	00101011	0.189206843	8	21-28
00000001	0.018177534	00010110	0.09908965	00101100	0.19324533		
00000010	0.019257477	00010111	0.099095282	00101101	0.195476764		
00000011	0.021555887	00011000	0.101533889	00101110	0.210145637		
00000100	0.025857471	00011001	0.106941659	00101111	0.217732068		
00000101	0.029991950	00011010	0.107888905	00110000	0.217801594		
00000110	0.031922630	00011011	0.121658454	00110001	0.224277071		
00000111	0.041819864	00011100	0.122020518	00110010	0.227712826		
00001000	0.042297798	00011101	0.124774041	00110011	0.230383067		
00001001	0.042659856	00011110	0.125654587	00110100	0.231237816		
00001010	0.044165572	00011111	0.133503860	00110101	0.236444933		
00001011	0.046191556	00100000	0.137546595	00110110	0.240904997		
00001100	0.047401462	00100001	0.137762893	00110111	0.244165287		
00001101	0.047554673	00100010	0.137868992	00111000	0.248628960		
00001110	0.054616615	00100011	0.138601716	00111001	0.257846170		
00001111	0.054791790	00100100	0.138724636	0011010	0.268438821		
00010000	0.066946258	00100101	0.152234013	00111011	0.269054732		
00010001	0.070684335	00100110	0.177123754	…………	…………		
00010010	0.082592727	00100111	0.178982479	…………	…………		
00010011	0.083873508	00101000	0.182141076	11111110	0.996156111		
00010100	0.087077220	00101001	0.182227506	11111111	0.99756035		

5.5.7 Representation of the momentum rate

The 29th and 38th genes represent the discrete value of momentum rate with which the network was trained. Table 6.

Table 6: Binary encoding of momentum rate value of the ANN

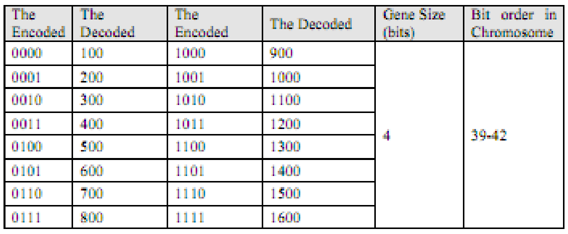


The Encoded	The Decoded	The Encoded	The Decoded	The Encoded	The Decoded	Gene Size (bits)	Bit order in Chromosome
0000000000	0.000341	0000010101	0.023632	0000101011	0.049632		
0000000001	0.001198	0000010110	0.024855	0000101100	0.05034		
0000000010	0.001301	0000010111	0.025151	0000101101	0.050646		
0000000011	0.001419	0000011000	0.026107	0000101110	0.051314		
0000000100	0.003394	0000011001	0.029332	0000101111	0.051436		
0000000101	0.004580	0000011010	0.030270	0000110000	0.051448		
0000000110	0.005834	0000011011	0.030385	0000110001	0.052192		
0000000111	0.007349	0000011100	0.032073	0000110010	0.053863		
0000001000	0.008648	0000011101	0.032940	0000110011	0.053978		
0000001001	0.009333	0000011110	0.033179	0000110100	0.055953		
0000001010	0.009759	0000011111	0.034866	0000110101	0.055976	10	29-38
0000001011	0.010979	0000100000	0.035423	0000110110	0.056343		
0000001100	0.011681	0000100001	0.036114	0000110111	0.056933		
0000001101	0.014362	0000100010	0.036382	0000111000	0.057340		
0000001110	0.015645	0000100011	0.036426	0000111001	0.057654		
0000001111	0.016675	0000100100	0.039184	0000011010	0.059031		
0000010000	0.017173	0000100101	0.043390	0000111011	0.059095		
0000010001	0.018613	0000100110	0.047078	…………	…………		
0000010010	0.019621	0000100111	0.047787	…………	…………		
0000010011	0.019765	0000101000	0.048739	1111111110	0.999329		
0000010100	0.020618	0000101001	0.049213	1111111111	0.999478		

5.5.8 Representation of the number of epochs

The 39th and 42th genes represent the discrete value of epochs with which the network was trained. Table 7

Table7 Binary encoding numbers epoch of the ANN



The Encoded	The Decoded	The Encoded	The Decoded	Gene Size (bits)	Bit order in Chromosome
0000	100	1000	900		
0001	200	1001	1000		
0010	300	1010	1100		
0011	400	1011	1200	4	39-42
0100	500	1100	1300		
0101	600	1101	1400		
0110	700	1110	1500		
0111	800	1111	1600		

5.6 Generation of initial population

An ANN-GA model was started with an initial population of elements which contains a predefined number of chromosomes (strings). To obtain ANN-GA model, the population size, mutation rate, and the crossover rate of the GA optimization parameters were set as 20, 0.05, and 0.9 respectively.

5.7 Neural network prediction

After the initial population has been generated, the population was fed into a trained ANN for prediction purpose. The input of the network was a set of initial process parameters generated by the GA optimization part.

5.8 Fitness evaluation

A fitness function was recognized to evaluate the fitness of individual population. In our case, the fitness of each individual string is the negative of mean square error (MSE). To keep the values as positive, the following formula was used:

$$\text{Fitness}= 10^6 - \frac{1}{n}\sum_{i=1}^{n}(y_i - y_t)^2 \qquad (3)$$

Where yt is target output and yi is the actual output.

5.9 Termination criteria

The evaluation processes will continue until some termination criteria are applied. In this case, the settling boundary and the maximum number of iteration are defined as the termination criteria in GA optimization.

5.10 Creation of a new population

In our model, new population is generated by applying reproduction operator (selection) and recombination operators (crossover and mutation). The operators are repeated until some conditions are satisfied.

5.11 Performance measurement

Mean square error (MSE) is used to evaluate ANN-GA performance. MSE can be calculated by equation 4, where T denotes the number of data patterns, Y_k is prediction output of kth pattern, and T_k is target output of kth pattern.

$$MSE = \frac{1}{T}\sum_{k=1}^{T}(Y_k - T_k) \qquad (4)$$

6. RESULTS & DISCUSSION

A new methodology that uses GA has been developed to determine optimum ANN parameters. The new ANN-GA model has been tested for predictions of three different dataset and the results were compared against those obtained by ANN using trial-and-error procedure.

The training, validation, and testing data for NN predictive model part were fed into GA part; the GA parameters that were used to predict the results are given in Table 7.

Table 8 GA parameters

parameters	value
Crossover probability	0.9
Mutation probability	0.05
Population size	20
Next generation	20

6.1 ANN-GA model prediction results for iC5 and nC5 of CRU debutanizer unit

The best solution found was the following string:

10 001001 01 1000000001 00000100 0000111110 0001

This is interpreted as:

A single hidden layer network with 15 neurons, logistic sigmoid activation function in hidden neurons and pure linear function in output neurons, trained with the Levenberg-Marquardt back-propagation algorithm that used a 0.519716 value as initial weight, by learning rate of 0.025857471, with a value for momentum term of 0.061401 after 200 epochs. This solution gave a MSE value of 0.0019 for iC5 and 0.0002 for nC5.

The second best solution found was the following string:

10 000100 10 1100011011 00001100 0001100000 0101

This is interpreted as:

A single hidden layer with 7 neurons, tangent sigmoid activation function in hidden neurons, and tangent sigmoid in output neurons, trained with Levenberg-Marquardt back-propagation algorithm that used a 0.784855 value as initial weight, by learning rate of 0.047401462, with a value for momentum term of 0.095949 after 600 epochs. This solution gave MSE value of 0.0019 for iC5 and 0.00027 for nC5.

6.2 ANN-GA model prediction results for XOR problem

The best solution found was the following string:

00 111100 10 0000011010 00000110 0000110001 0011

That is interpreted as:

two hidden layer NN with 8 neurons in the first hidden layer and 5 neurons in the second hidden layer, tangent sigmoid activation function in first hidden neurons, tangent sigmoid function in second hidden neurons, and pure linear function in output neurons, trained with the Steepest Decent back-propagation algorithm that used a 0.059031value as initial weight, by learning rate of 031922630, with a value for momentum term of 052192 after 400 epochs. This solution gave a MSE value of 0.0002 for XOR output.

The second best solution found was the following string:

10 001101 11 0000110010 00111001 0001000000 0001

That is interpreted as:

A single hidden layer with 17 neurons, tangent sigmoid activation function in first hidden neurons, and pure linear function in output neurons, trained with Levenberg-Marquardt back-propagation algorithm that used a 0.053863 value as initial weight, by learning rate of 0.257846170, with a value for momentum term of 0.024855after 200 epochs. This solution gave a MSE value of 0.0004 for XOR output.

7. COMPARATIVE STUDY

The performance of the proposed ANN-GA approach is compared to the existing five approaches in terms of prediction effectiveness; the results are summarized in Figure 6.

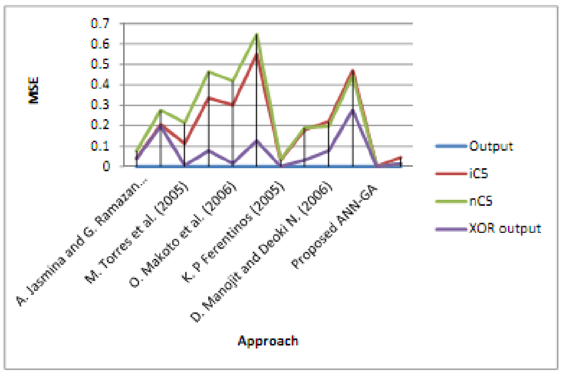


Fig 6: Comparison of proposed approach with other approaches in terms of prediction effectiveness (MSE)

A comparison of the results achieved by the proposed ANN-GA approach and by the other approaches in terms of simulation time clearly indicates that the proposed ANN-GA approach has achieved acceptable degree of success. Figure 7 compares the simulation time of the proposed approach and other approaches

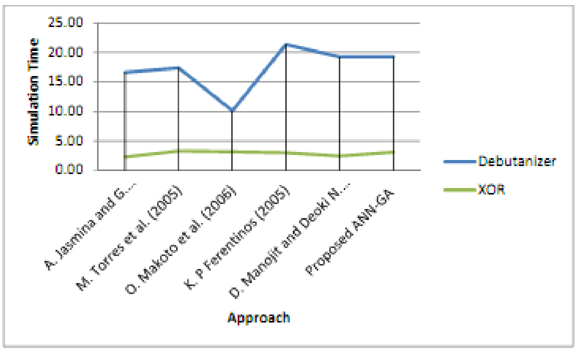


Fig 7: Comparison of proposed approach with other approaches in terms of simulation time

7. CONCLUSION & FUTURE DIRECTIONS

In this paper, GA to determination of optimum ANN parameters is presented. The target is to design ANN with satisfactory performance and less human-dependent. Different datasets have been collected, treated, analyzed, and scaled. The GA has been applied to the ANN model so as to get optimal ANN parameters of design and training. Neural network and genetic algorithm MATLAB toolboxes have been used to obtain the results which proofed that the new model can optimize ANN parameters precisely and effectively. As a future direction of this work, the GA will be applied to determine optimum ANN parameters within a wide search space i.e. find optimal ANN architecture from three hidden layers, optimal minimization algorithm from six different types, and study the impact of search space on training time, performance, and complexity.

8. REFERENCES

[1] H. Patrick C.L. (2007). “Application of artificial neural networks to the prediction of sewing performance of fabrics”, International Journal of clothing and Technology, Vol. 19, No.5, pp. 291318.

[2] F. Fred F., G. James D., and L. Juliet N. (2000). “Predicting temperature profiles in producing oil wells using artificial neural networks”, Engineering Computation, Vol. 17, No. 6, pp. 0264-4401.

[3] K. Okyay, Springer, 2003. “Artificial Neural Networks and Neural Information”, ISBN 3540404082.

[4] D. Satyanarayana, K. Kamarajan, and M. Rajappan (2005). “Genetic Algorithm Optimized Neural Networks Ensemble for Estimation of Mefenamic Acid and Paracetamol in Tablets”, Genetic Algorithm Optimized Neural Networks Ensemble, Acta Chim. Slov., Vol. 52, pp. 440–449.

[5] M. Izadifar, M. Zolghadri Jahromi (2007). “Application of genetic algorithm for optimization of vegetable oil hydrogenation process”, Journal of Food Engineering, Vol. 78, Issue 1, pp. 1-8.

[6] F. Konstantinos P. (2005). “Biological engineering applications of feedforward neural networks designed and parameterized by genetic algorithms”, Neural Networks, Vol. 18, Issue 7, pp. 934-950.

[7] M. Majors, J. Stori, C. Dong-I (2002), “Neural network control of automotive fuel-injection systems”, Control Systems Magazine, IEEE, Vol. 14, Issue 3, pp. 31-36.

[8] C. Arzum E., K. Yalcin (2007). “Evaluating and forecasting banking crises through neural network models: An application for Turkish banking sector”, Expert Systems with Applications, Vol. 33, Issue 4, pp. 809-815.

[9] L. Bor-Ren and R.G. Hoft (2003). “Neural networks and fuzzy logic in power electronics”, Control Engineering Practice, Vol. 2, Issue 1, pp. 113-121.

[10] Z. Xiaotian, X. Hong Wang, Li, and L. Huaizu Li (2008). “Predicting stock index increments by neural networks: The role of trading volume under different horizons”, Expert Systems with Applications, Vol. 34, Issue 4, pp. 3043-3054.

[11] G.R. Cheginia, J. Khazaeia, B. Ghobadianb, and A.M. Goudarzic Constructing ANN (2008). “Prediction of process and product parameters in an orange juice spray dryer using artificial neural networks”, Journal of Food Engineering, Vol. 84, Issue 4, pp 534-543.

[12] N. Perambur S. and A. Preechayasomboon (2002). “Development of a neuroinference engine for ADSL modem applications in telecommunications using an ANN with fast computational ability”, Neurocomputing, Vol. 48, Issues 1-4, pp. 423-441.

[13] F. Fred F., G. James D., and L. Juliet N. (2000). “Predicting temperature profiles in producing oil wells using artificial neural networks”, Engineering Computation, Vol. 17, No. 6, pp. 0264-4401.

[14] N. Huang, K.K. Tan, and T.H. Lee (2008), “Adaptive neural network algorithm for control design of rigid-link electrically driven robots”, Neurocomputing, Vol. 71, Issues 4-6, pp 885-894.

[15] D.E. Goldberg (1989). In: (second ed.), Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA.

[16] S.L. Mok, C.K. Kwong, and W.S. Lau (2001). “A Hybrid Neural Network and Genetic Algorithm Approach to the Determination of Initial Process Parameters for Injection Moulding”, The International Journal of Advanced Manufacturing Technology, Vol.18, pp. 404-409.

[17] H. Paul S., G. Ben S., T. Thomas G., and W. Robert S. (2004). “Use of genetic algorithms for neural networks to predict community-acquired pneumonia”, Artificial Intelligence in Medicine, Vol. 30, Issue 1, pp. 71-84.

[18] F. Konstantinos P. (2005). “Biological engineering applications of feedforward neural networks designed and parameterized by genetic algorithms”, Neural Networks, Vol. 18, Issue 7, pp.934-950.

[19] S.S. Panda, D. Chakraborty, and S.K. Pal (2008). “Flank wear prediction in drilling using back propagation neural network and radial basis function network”, Applied Soft Computing, Issue 8, pp. 858–871.

[20] J.P. Marques de Sá, Joaquim P. Marques de Sa, Joaquim P. Marques de Sā (2007). “Applied Statistics Using SPSS, STATISTICA, MATLAB and R”, ISBN 3540719717, pp. 78-83, Springer

[21] E.A. Osman, M.A. Ayoub, and M.A. Aggour (2005). “Artificial Neural Network Model for Predicting Bottomhole Flowing Pressure in Vertical Multiphase Flow”, SPE Middle East Oil and Gas Show and Conference.

[22] D.E. Rumelhart, G.E. Hinton, and R.J. Williams (1986). “Learning internal representation by back propagating errors”, Nature 323, pp. 533–536.